Outils - Mise en oeuvre de DocFX

[Mise à jour le 18/8/2023]

- **Source**
  - Cet article a été écrit à partir de la documentation du site **DocFx**

> Les différentes parties doivent être abordées dans l'ordre en première lecture.

A. Qu'est-ce que DocFX ?

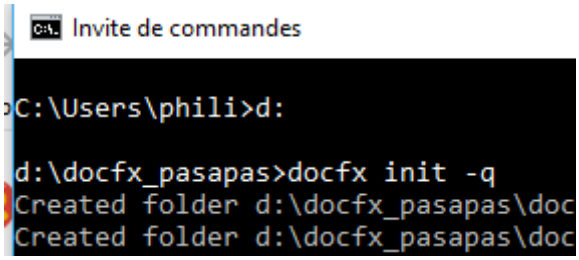
DocFX est un **générateur de documentation API pour .NET**, qui prend actuellement en charge C# et VB. Il génère une documentation de référence API à partir de commentaires triples dans votre code source. Il vous permet également d'utiliser des fichiers **Markdown**[1)] pour créer des rubriques supplémentaires telles que des didacticiels et des procédures, et de personnaliser la documentation de référence générée. DocFX crée un site Web HTML statique à partir de votre code source et de vos fichiers Markdown, qui peut être facilement hébergé sur tous les serveurs Web (par exemple, **github.io**). De plus, DocFX vous permet de personnaliser la présentation et le style de votre site Web à l'aide de modèles. Si vous souhaitez créer votre propre site Web avec vos propres styles, vous pouvez suivre la procédure de création d'un modèle personnalisé.

B. DocFX étape par étape

1. Générer un site Web de documentation

Nous allons nous familiariser avec le principe général d'organisation des documents dans docFX pour générer un site statique.

- **Étape 1. Configurer DocFX**
  Il y a trois manières d'installer DocFX. Dans ce qui suit, nous utilisons directement l'exécutable ***docfx.exe***.
  1. Télécharger docfx.zip et le dézipper dans D:\docfx\
  2. Sous Windows, ajouter le chemin D:\docfx\ aux variables d'environnement (PATH) pour que la commande docfx soit accessible dans tous les dossiers.
- **Étape 2. Initier un projet DocFX**
  1. Créer un nouveau répertoire. Par exemple *D:\docfx_pasapas*.
  2. Ouvrir la ligne de commande dans ce nouveau dossier.
  3. Exécuter la commande **`docfx init -q`**. Cette commande génère un dossier *docfx_project* contenant le fichier ***docfx.json*** par défaut. docfx.json est le fichier de configuration que docfx utilise pour générer la documentation.

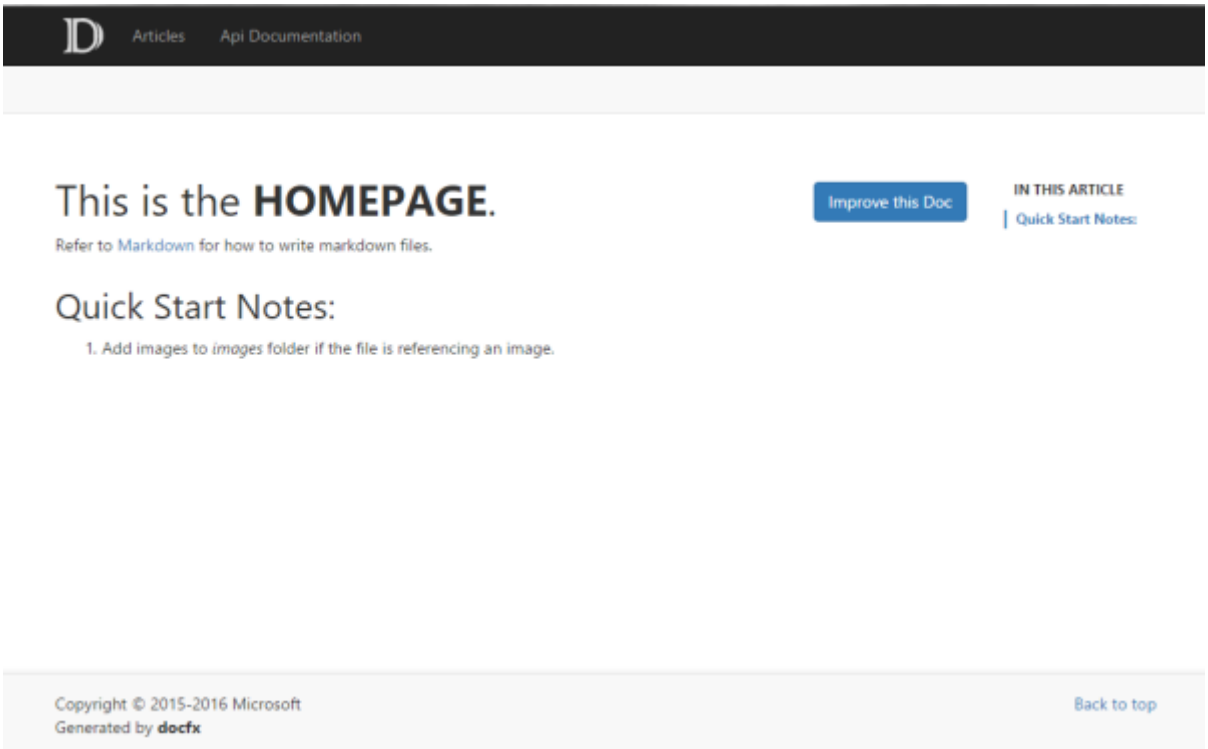


L'option **-q** signifie que le projet est généré en utilisant les valeurs par défaut, vous pouvez également essayer docfx init et suivre les instructions pour fournir vos propres paramètres.

- **Étape 3. Construire le site Web**
  Exécuter la commande **`docfx`** *`docfx_project/docfx.json`*. Notez qu'un nouveau sous-dossier **_site** est généré dans ce dossier. C'est là que le site Web statique est généré.
- **Étape 4. Prévisualiser le site Web**
  Le site Web statique généré peut être publié sur les pages **GitHub**, sur les sites Web Azure ou sur vos propres services d'hébergement, sans aucune modification supplémentaire. Vous pouvez également exécuter la commande **`docfx serve`** *`docfx_project/_site`* pour prévisualiser le site Web localement.

Si le port **8080** n'est pas utilisé, docfx hébergera *_site_* sous **http://localhost:8080**. Si 8080 est utilisé, vous pouvez utiliser **`docfx serve _site -p`** <port> pour changer le port.

La page devrait ressembler à ceci.



- **Étape 5. Ajouter des articles au site Web**

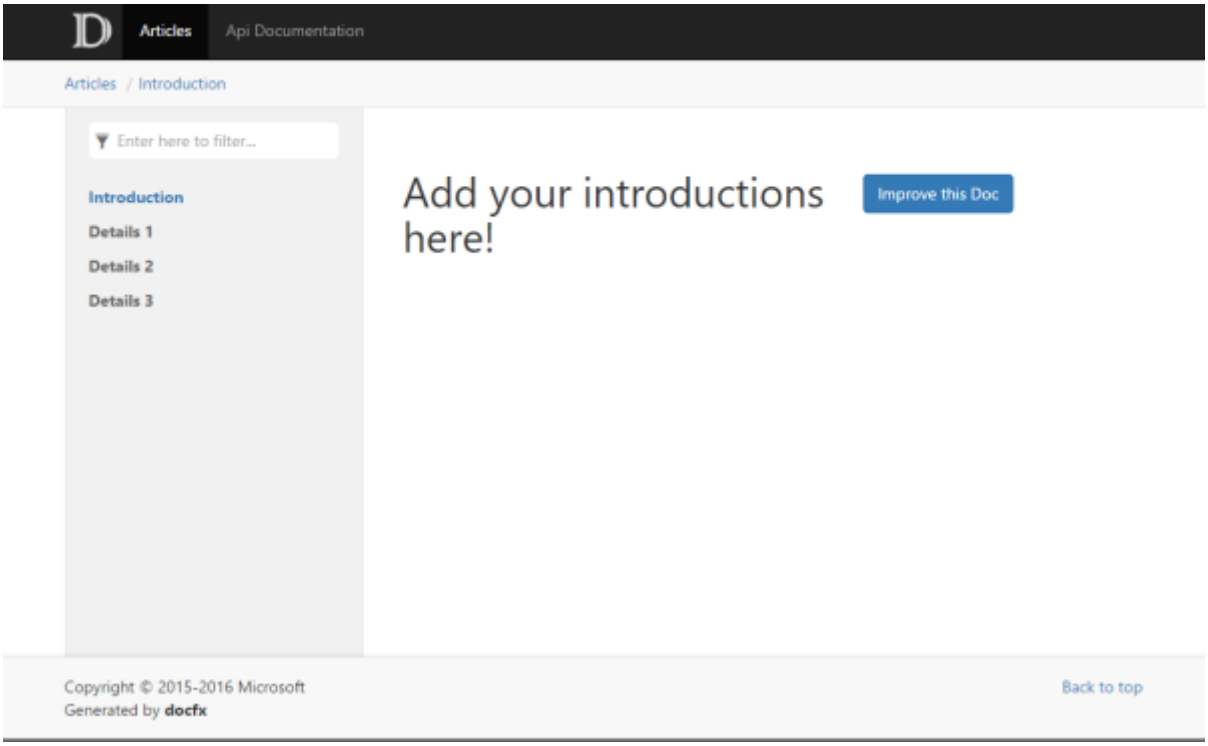
1. Pour ajouter des articles au site, il suffit de placer des fichiers au format markdown **.md** dans le sous-dossier ***articles***. Ajoutons par exemple les fichiers *details1.md*, *details2.md*, *details3.md*. Si les fichiers font référence à des ressources, placez-les dans le dossier images.
2. Afin d'organiser ces articles, ajoutons des références à ces fichiers dans le fichier ***toc.yml*** contenu dans le sous-répertoire ***articles***. Le contenu de *toc.yml* doit être complété comme ci-dessous:

```
- name: Introduction
  href: intro.md
- name: Details 1
  href: details1.md
- name: Details 2
  href: details2.md
- name: Details 3
  href: details3.md
```

L'organisation des dossiers est maintenant :

```
|- index.md
|- toc.yml
|- articles
|    |- intro.md
|    |- details1.md
|    |- details2.md
|    |- details3.md
|    |- toc.yml
|- images
    |- details1_image.png
```

3. Exécuter à nouveau les commandes des étapes 3 et 4 puis **cliquer** sur ***Article*** pour que le site ressemble à ceci :



Conclusion

Dans cette procédure, nous avons construit un site Web à partir d'un ensemble de fichiers markdown. Ces fichiers .md sont la **documentation conceptuelle**. Dans le paragraphe suivant, nous allons apprendre à ajouter la documentation d'une API [2)] à notre site Web. La documentation de l'API sera automatiquement extraite du code source .NET[3)]. Dans une série de procédures avancées, nous verrons d'autres concepts de docFX, tels que les références croisées entre articles, les références externes à d'autres documentations, etc.

---

2. Ajouter la documentation d'une API

Dans cette partie, nous allons apprendre à créer un site Web à partir du code source d'un projet .NET, il s'agit de la **Documentation de l'API**. Nous allons également intégrer la documentation conceptuelle et la documentation de l'API dans un site Web unique, de manière à pouvoir naviguer de "Conceptuel" à "API", ou d'"API" à "Conceptuel".

- **Étape 1. Ajouter un projet C#** 

  1. Créer un sous-dossier ***src*** sous D:\docfx_pasapas\docfx_project.
  2. Ouvrir **Visual Studio Community** 2015 (ou une version supérieure) et créer, dans le dossier *src*, une bibliothèque de classes C# que vous nommerez ***HelloDocfx*** .
  3. Remplacer le contenu du fichier *Class1.cs*, par celui du fichier téléchargeable ici.

- **Étape 2. Générer les métadonnées du projet C#**
  1. Se placer dans le dossier *docfx_project*
  2. Exécuter la commande **`docfx metadata`** dans le dossier **D:\docfx_pasapas\docfx_project**. La commande `docfx metadata` lit la configuration dans la section metadata du fichier *docfx.json*. La partie ["src/**.csproj"] située dans *metadata/src/files* demande à docFX de rechercher tous les *csproj* du sous-dossier *src* pour générer les métadonnées.

```
"metadata": [
    {
      "src": [
        {
          "files": [
            "src/**.csproj"
          ],
          "exclude": [
            "**/bin/**",
            "**/obj/**",
            "_site/**"
          ]
        }
      ],
      "dest": "api"
    }
  ]
```

Cela génère plusieurs fichiers **YAML**[4] dans le dossier ***api***. Un fichier YAML contient le modèle de données extrait du fichier de code source C#. YAML est le format de métadonnées utilisé dans docFX. La spécification de métadonnées générales définit le schéma général et la spécification de métadonnées .NET définit le schéma de métadonnées pour les langages .NET pouvant être consommés par docFX.

```
|- HelloDocfx.Class1.InnerClass.yml
|- HelloDocfx.Class1.yml
|- HelloDocfx.yml
|- toc.yml
```

> Notez également que si vos csproj sont situés en dehors de votre répertoire docFX et que vous devez utiliser ../, vous devrez compléter la propriété src.

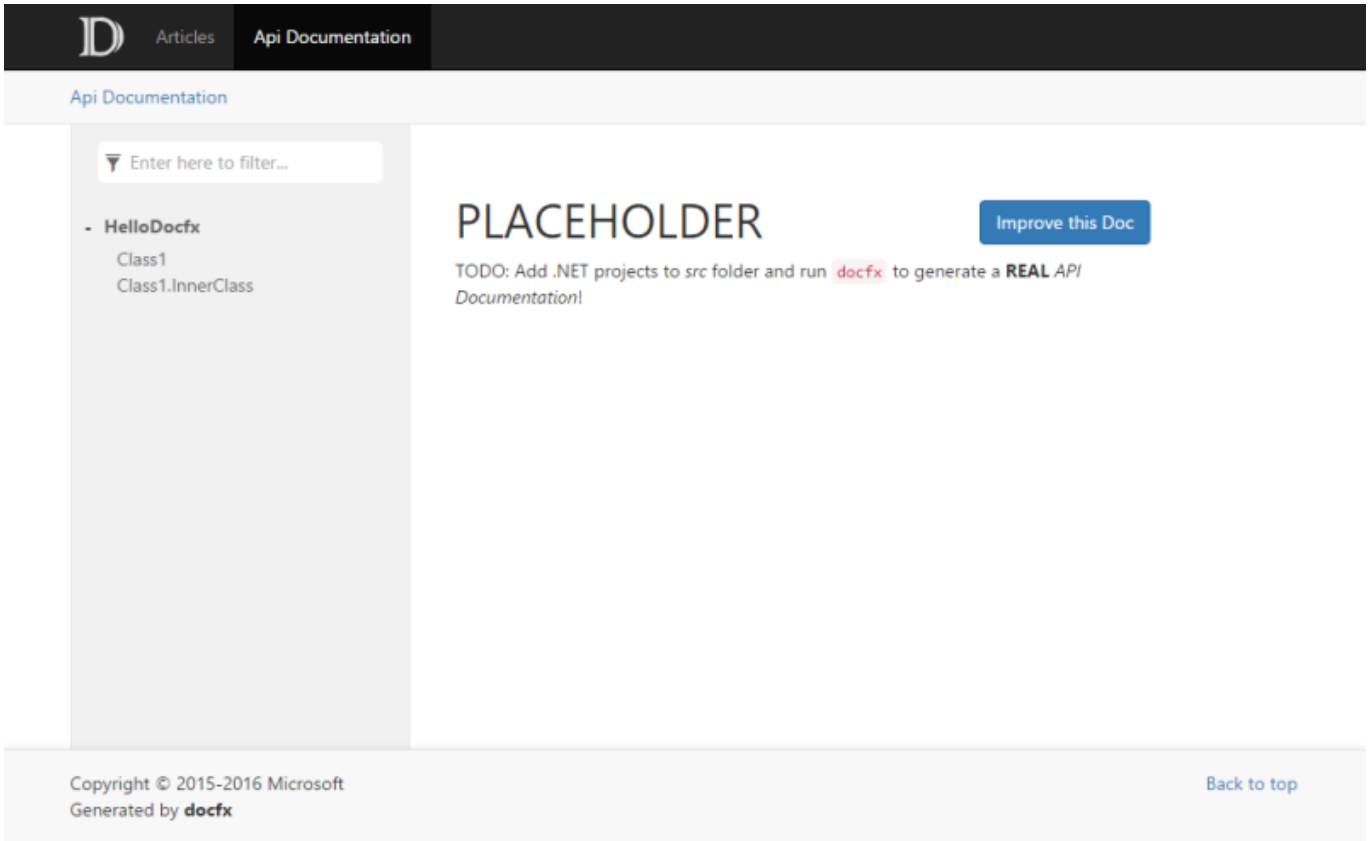
```
"metadata": [
    {
      "src": [
        {
          "src": "../../",
          "files": [
            "somewhere/src/**.csproj"
          ],
```

```
        "exclude": [
          "**/bin/**",
          "**/obj/**",
          "_site/**"
        ]
      }
    ],
    "dest": "api"
  }
]
```

- **Étape 3. Construire et prévisualiser le site**

Lancer la commande **`docfx`**. Cette commande lit le fichier *docfx.json* et exécute, une par une, les sous-commandes. Notre fichier *docfx.json* définit les `metadata` et le `build`. En exécutant la commande docfx, nous construisons le site Web.

Exécutez la commande **`docfx serve _site`** puis **cliquer** sur **API Documentation**. Le site devrait ressembler à :



- **Étape 4. Ajouter une autre documentation d'API**
  1. Créer un autre sous-dossier ***src2*** sous D:\docfx_pasapas\docfx_project. Outre la génération de documentation API à partir de fichiers de projet, docFX peut générer de la documentation directement à partir du code source.
  2. Télécharger le fichier Class2.cs, le dézipper et le placer dans *src2*
  3. Compléter la section *metadata* du fichier *docfx.json* comme ci-dessous :

```
"metadata": [
   {
```

```
      "src": [
        {
          "files": [
            "src/**.csproj"
          ]
        }
      ],
      "dest": "api",
      "disableGitFeatures": false,
      "disableDefaultFilter": false
    },
    {
      "src": "src2/**.cs",
      "dest": "api-vb"
    }
  ],
```

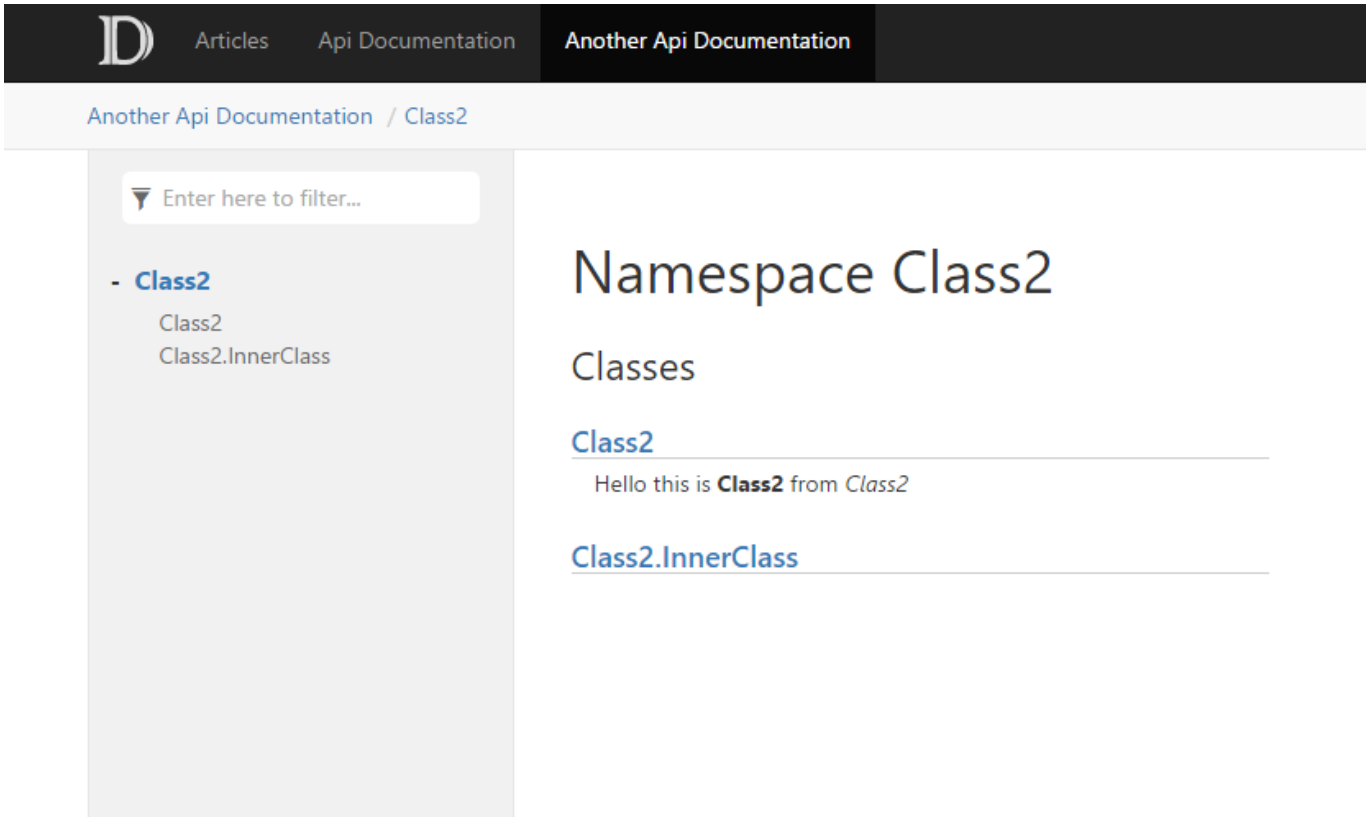
4. Cela signifie que les fichiers de métadonnées YAML pour "src2/**.cs" sont générés dans le dossier “api-vb”. Incluons également les fichiers YAML générés dans la section build :

```
"build": {
    "content": [
      {
        "files": [
          "api-vb/**.yml"
        ]
      }
      ...
```

5. Pour qu'il soit organisé et affiché sur le site Web, nous devons également modifier le fichier D:\ docfx_walkthrough\docfx_project**toc.yml**. Ne pas oublier d'ajouter une barre oblique **/** pour la valeur de **href**.:

```
- name: Articles
  href: articles/
- name: Api Documentation
  href: api/
  homepage: api/index.md
- name: Another Api Documentation
  href: api-vb/
```

6. Exécuter la commande **`docfx - -serve`** puis **cliquer** sur **Another Api Documentation**. Le site devrait ressembler à :



- **Étape 5. Combinez les informations conceptuelles et de référence dans la barre de navigation gauche**
  La barre de **navigation de gauche** peut contenir des liens vers des **informations conceptuelles** (vue d'ensemble, démarrage, etc.) et des **informations de référence**.
  1. Remplacer le contenu du fichier **toc.yml**, à la racine, par le texte ci-dessous. (Il détermine le contenu de la barre de menus horizontale principale.)

```
- name: Home
  href: index.md
- name: Articles
  href: articles/
  homepage: api/index.md
- name: API Documentation
  href: obj/api/
- name: REST API
  href: restapi/
```

  2. Ajouter un nouveau dossier à la racine (par exemple, **fusion**).
  3. À l'intérieur de fusion, ajoutez ***toc.yml*** et dans *toc.yml*, ajouter le texte ci-dessous:

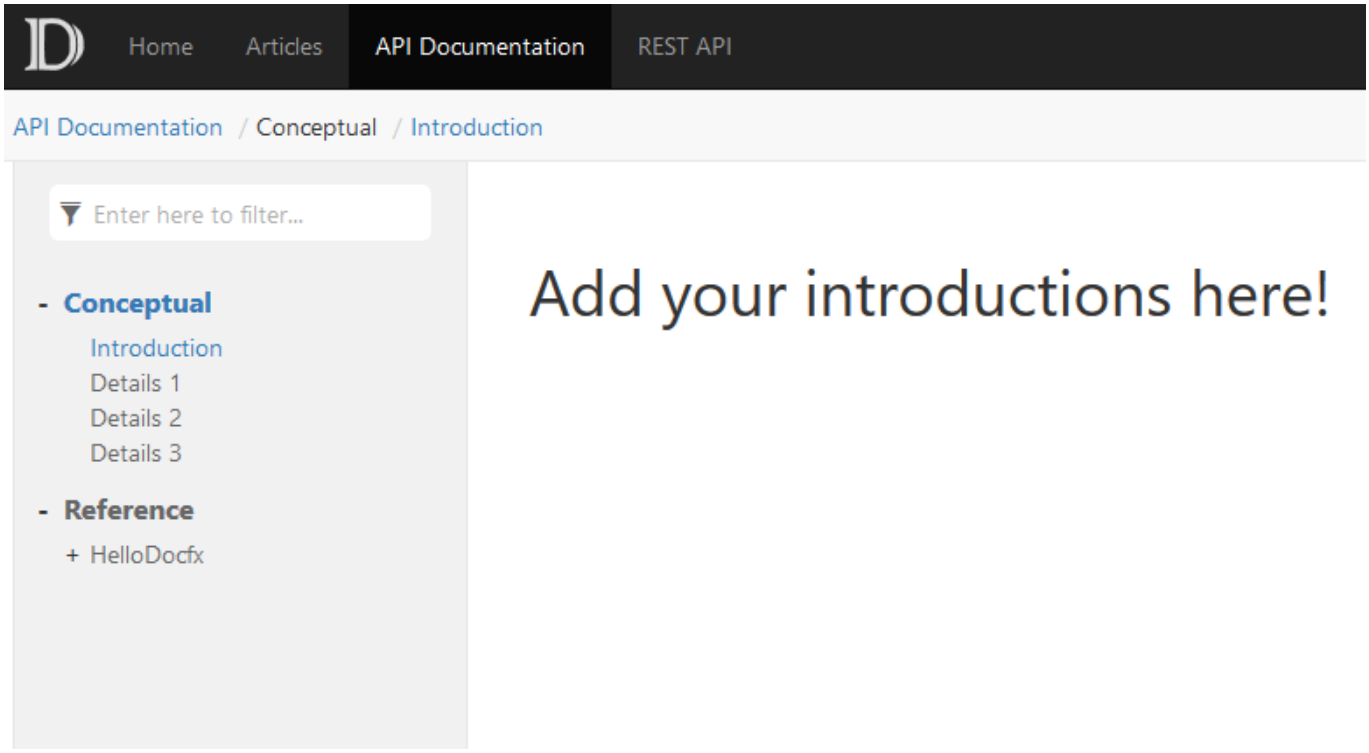
```
- name: Conceptual
  href: ../articles/toc.yml
- name: Reference
  href: ../obj/api/toc.yml
```

  4. Dans le fichier *toc.yml* à la racine, remplacez ../obj/api/toc.yml par le chemin vers fusion :

```
- name: Home
  href: index.md
- name: Articles
```

```
  href: articles/
  homepage: api/index.md
- name: API Documentation
  href: fusion/
- name: REST API
  href: restapi/
```

5. Exécuter la commande **`docfx - -serve`** puis cliquer sur **Api Documentation**. Le site devrait ressembler à :



3. Générer la documentation au format pdf

A l'étape précédente, nous avons créé un site Web contenant à la fois la documentation conceptuelle et la documentation API. Dans cette partie, nous allons générer cette documentation au format PDF.

docfx_project obtenu au paragraphe précédent a été réduit a la structure ci desous. A télécharger ici.

```
|- articles
|- images
|- src
|- src2
|- index.md
|- toc.yml
|- docfx.json
```

- **Étape 0. Installer les prérequis**

Nous utiliserons wkhtmltopdf[5] pour générer des PDF. Télécharger l'exécutable wkhtmltopdf et l'installer.

- **Étape 1. Ajouter un fichier toc.yml spécifique aux PDF**

Chaque fichier TOC génère le fichier PDF correspondant, TOC est également utilisé pour la page de couverture du PDF, nous créons donc un fichier toc.yml spécifique au PDF dans un nouveau dossier pdf, en utilisant TOC Include pour inclure le contenu d'autres fichiers TOC.

```
- name: Articles
  href: ../articles/toc.yml
- name: Api Documentation
  href: ../api/toc.yml
- name: Another Api Documentation
  href: ../api-vb/toc.yml
```

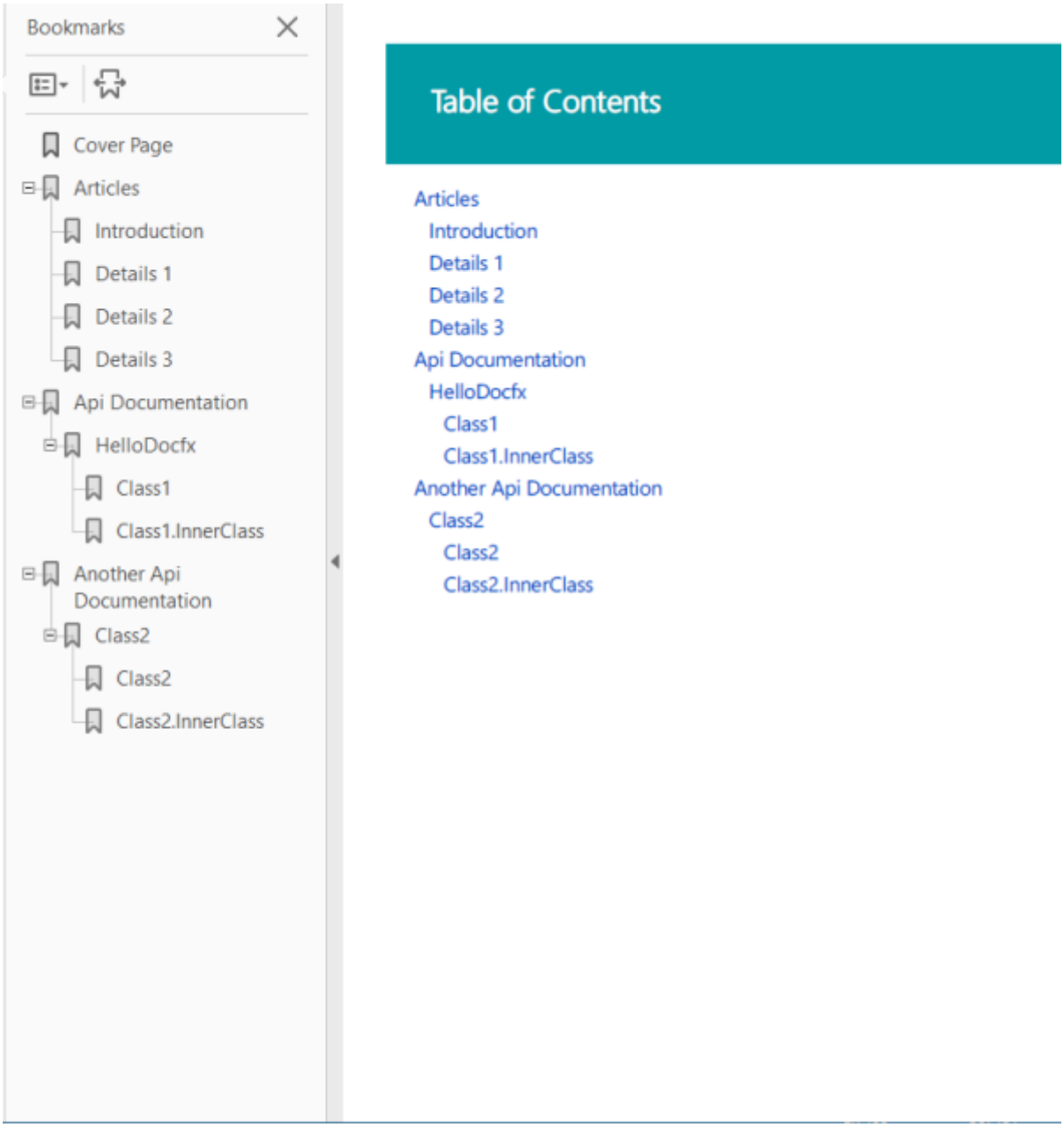
- **Étape 2. Ajouter une section pdf dans docfx.json**

Les paramètres ressemblent à ceux de la section “construction”, bien qu'on utilise un modèle différent (le modèle intégré est pdf.default), avec une autre destination. Nous excluons les fichiers TOC car chaque fichier TOC génère un fichier PDF:

```
...
  "pdf": {
    "content": [
      {
        "files": [
          "api/**.yml",
          "api-vb/**.yml"
        ],
        "exclude": [
          "**/toc.yml",
          "**/toc.md"
        ]
      },
      {
        "files": [
          "articles/**.md",
          "articles/**/toc.yml",
          "toc.yml",
          "*.md",
          "pdf/*"
        ],
        "exclude": [
          "**/bin/**",
          "**/obj/**",
          "_site_pdf/**",
          "**/toc.yml",
          "**/toc.md"
        ]
      },
      {
        "files": "pdf/toc.yml"
      }
```

```
    ],
    "resource": [
      {
        "files": [
          "images/**"
        ],
        "exclude": [
          "**/bin/**",
          "**/obj/**",
          "_site_pdf/**"
        ]
      }
    ],
    "overwrite": [
      {
        "files": [
          "apidoc/**.md"
        ],
        "exclude": [
          "**/bin/**",
          "**/obj/**",
          "_site_pdf/**"
        ]
      }
    ],
    "dest": "_site_pdf"
  }
```

Exécutons maintenant la commande **`docfx`**.Vous trouverez le fichier pdf *walkthrough3_pdf.pdf* généré dans le dossier _site_pdf:



- **Etape 3 facultative. Activer les plugins**

Si vous souhaitez également utiliser des plugins avec pdf, vous devez ajouter un noeud de modèle à la section pdf. Il doit commencer par le fichier pdf.template suivi du chemin d'accès aux plugins que vous souhaitez utiliser :

```
"template": [
  "pdf.default",
  "pluginPackages/rest.tagpage.2.31.0/content"
],
```

- **4. Personnaliser son site**

Voir la page consacrée à cette rubrique ici.

1)

Markdown est un langage de balisage léger créé en 2004 par John Gruber avec Aaron Swartz. Son but

est d'offrir une syntaxe facile à lire et à écrire.
2)

API : Application Programming Interface
3)

Microsoft .NET
4)

YAML, Acronyme de Yet Another Markup Language dans sa version 1.0, il devient l'acronyme récursif de YAML Ain't Markup Language (« YAML n'est pas un langage de balisage ») dans sa version 1.1, est un format de représentation de données par sérialisation Unicode.
5)

wkhtmltopdf est un outil en ligne de commande open source (LGPLv3) pour le rendu HTML en PDF